EEL 5722C
Field-Programmable Gate Array Design

Lecture 19: Hardware-Software Co-Simulation*

Prof. Mingjie Lin

UCF
Stands For Opportunity

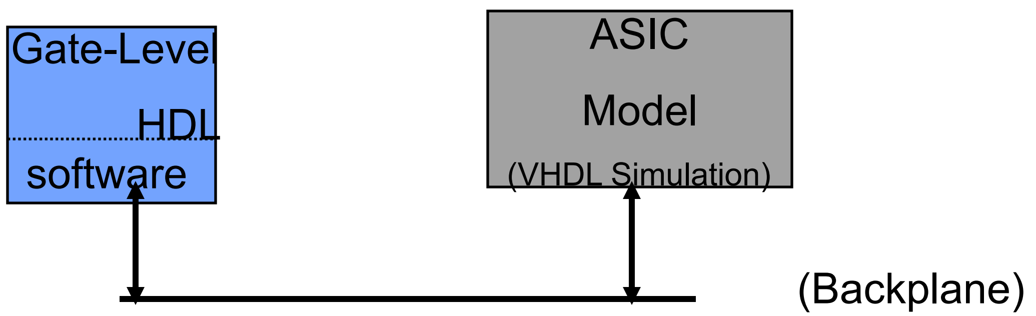
* Rabi Mahapatra, CpSc489

How to cosimulate?

- *How to simulate hardware components of a mixed hardware-software system within a unified environment?*
  - This includes simulation of the hardware module, the processor, and the software that the processor executes.
- *How to simulate hardware and software at same time?*
- What are various challenges?
  - Software runs faster than hardware simulator. How to run the system simulation fast keeping the above synchronized?
  - Slow models provide detailed and accurate results than fast models. How to balance these effects?
  - Use of different platforms for simulations.

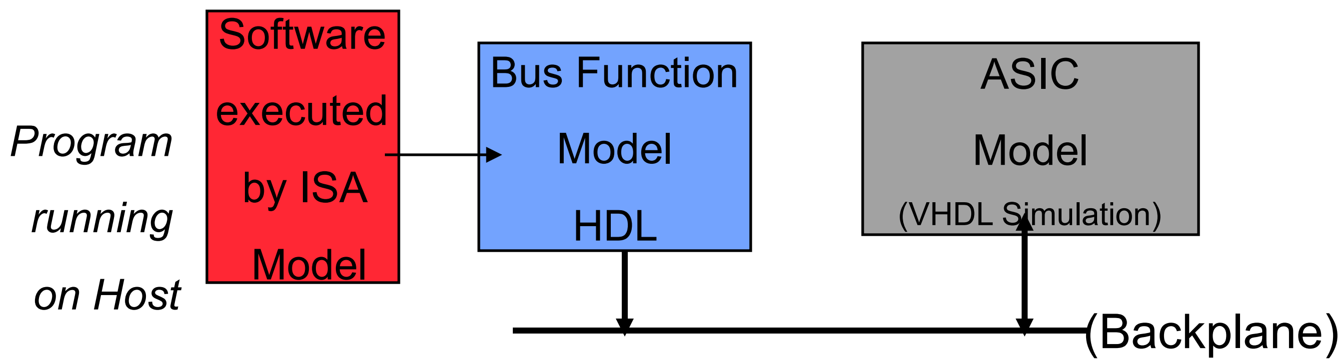
Some basic approaches

- Detailed Processor Model:
  - processor components(memory, datapath, bus, instruction decoder etc) are discrete event models as they execute the embedded software.
  - Interaction between processor and other components is captured using native event-driven simulation capability of hardware simulator.
  - Gate level simulation is extremely slow (~tens of clock cycles/sec), behavioral model is ~hundred times faster. Most accurate and simple model



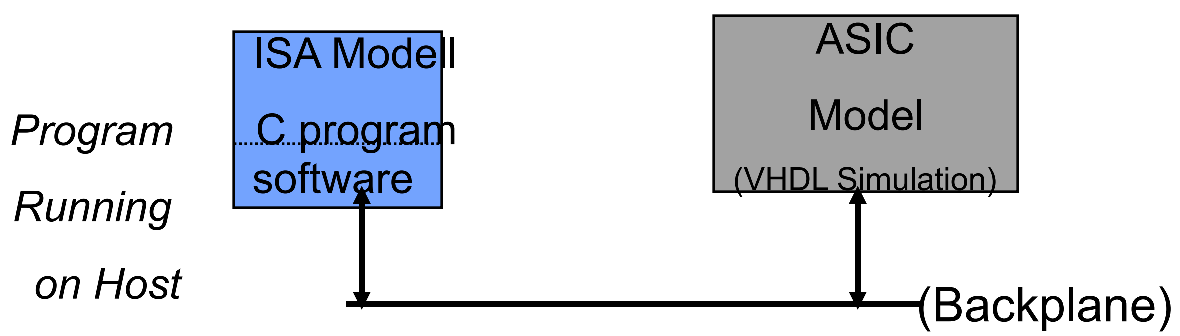
Some basic approaches

- Bus Model (Cycle based simulator):
  - Discrete-event shells that only simulate activities of bus interface without executing the software associated with the processor. Useful for low level interactions such as bus and memory interaction.
  - Software executed on ISA model and provide timing information in clock cycles for given sequence of instructions between pairs of IO operation.
  - Less accurate but faster simulation model.



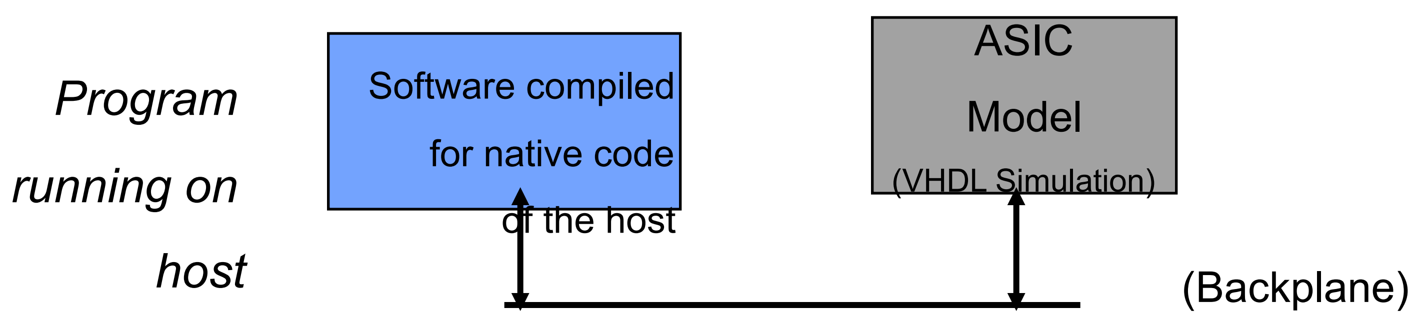
Some basic approaches

- Instruction Set Architecture Model:
  - ISA can be simulated efficiently by a C program. C program is an interpreter for the embedded software.
  - No hardware mode. Software executed on ISA model. Execution on ISA model provides timing (clock) details of the cosimulation.
  - Can be more efficient than detailed processor modeling because internals of the processor do not suffer the expense of discrete-event scheduling.



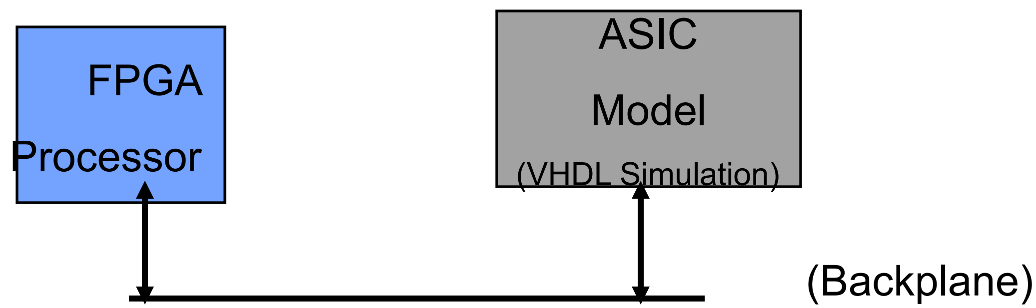
Some basic approaches

- Compiled Model:
  - very fast processor models are achievable in principle by translating the executable embedded software specification into native code for processor doing simulation. (Ex: Code for programmable DSP can be translated into Sparc assembly code for execution on a workstation)
  - No hardware, software execution provides timing details on interface to cosimulation.
  - Fastest alternative, accuracy depends on interface information.



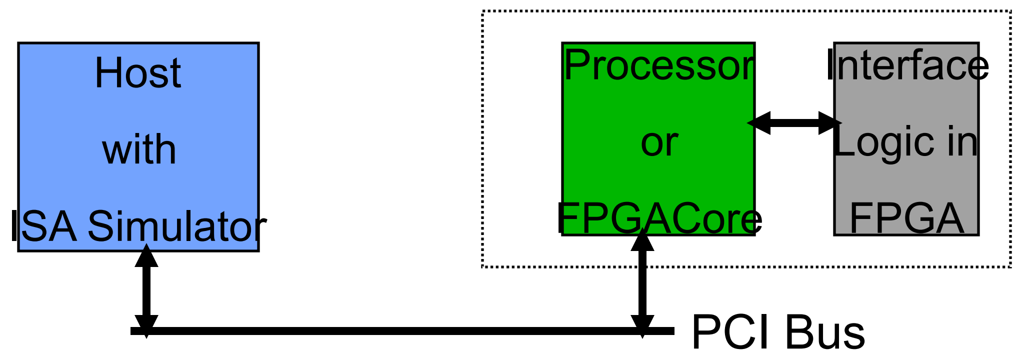
Some basic approaches

- Hardware Model:
  - If processor exists in hardware form, the physical hardware can often be used to model the processor in simulation. Alternatively, processor could be modeled using FPGA prototype. (say using Quickturn)
  - Advantage: simulation speed
  - Disadvantage: Physical processor available.



A New Approach

- This is a combined HW/SW approach. The host is responsible of having OS, some applications and might have superset simulating environment (RSIM, SIMICS, SIMOID).
- Use of fast backplane (PCI) for communication. Real processor or processor core in FPGA as hardware model, and ASIC/FPGA for interface and interconnection for hardware modeler.
- Good for fast complex architecture simulations including multiprocessor.



Domain coupling

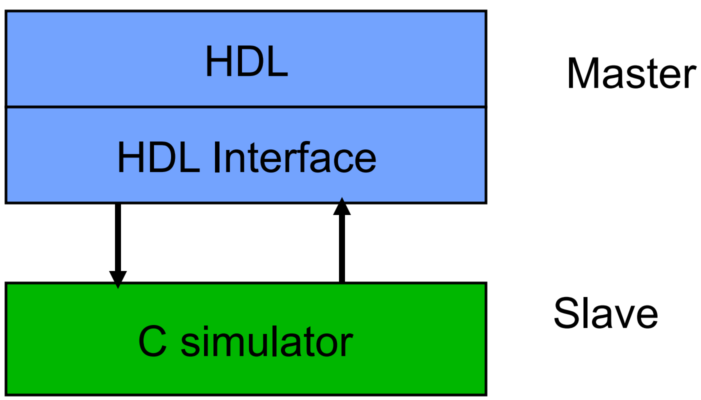
- In four out of six approaches, we have host that run software and required to interact with hardware model or simulator.
- Difficulties:
  – providing timing information across the boundaries
  – coupling two domains with proper synchronization

Migration across cosimulation

- Consider the system simulation at different levels of abstraction throughout the design process:
  - In the beginning of design process, hardware synthesis is not available. Hence use functional model to study the interaction between HW and SW.
  - As design progress with more implementations, replace functional model of hardware by netlist level.
  - Once detail operation of hardware is verified, swap back the high level description of HW design to gain simulation speed.
- The cosimulation environment should have this migration support across the levels of abstraction.
- Off-the-shelf Components: design is not a part of the current design process. Functional model is enough, no need to know internal details.

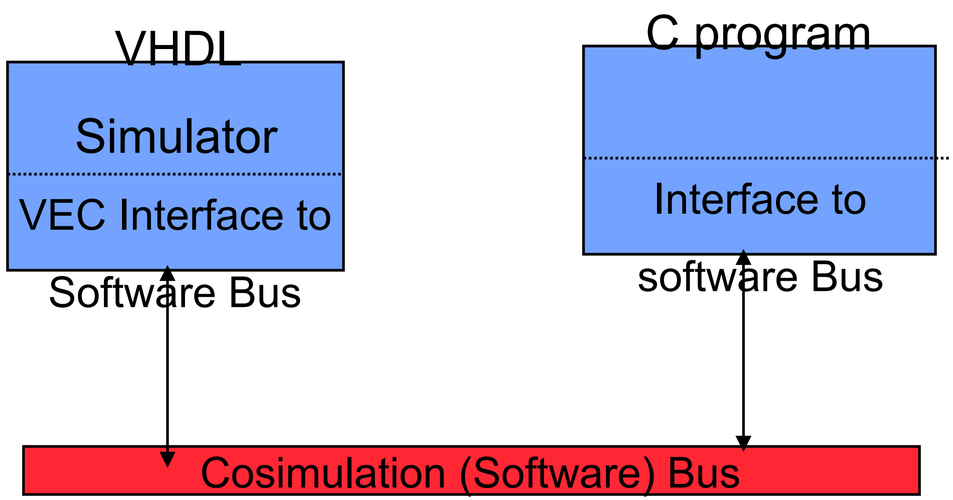
Master slave cosimulation

- One master simulator and one or more slave simulators: slave is invoked from master by procedure call.
- The language must have provision for interface with different language
- Difficulties:
  - No concurrent simulation possible
  - C procedures are reorganized as C functions to accommodate calls



Distributed cosimulation

- Software bus transfers data between simulators using a procedure calls based on some protocol.
- Implementation of System Bus is based on system facilities (Unix IPC or socket). It is only a component of the simulation tool.
- Allows concurrency between simulators.

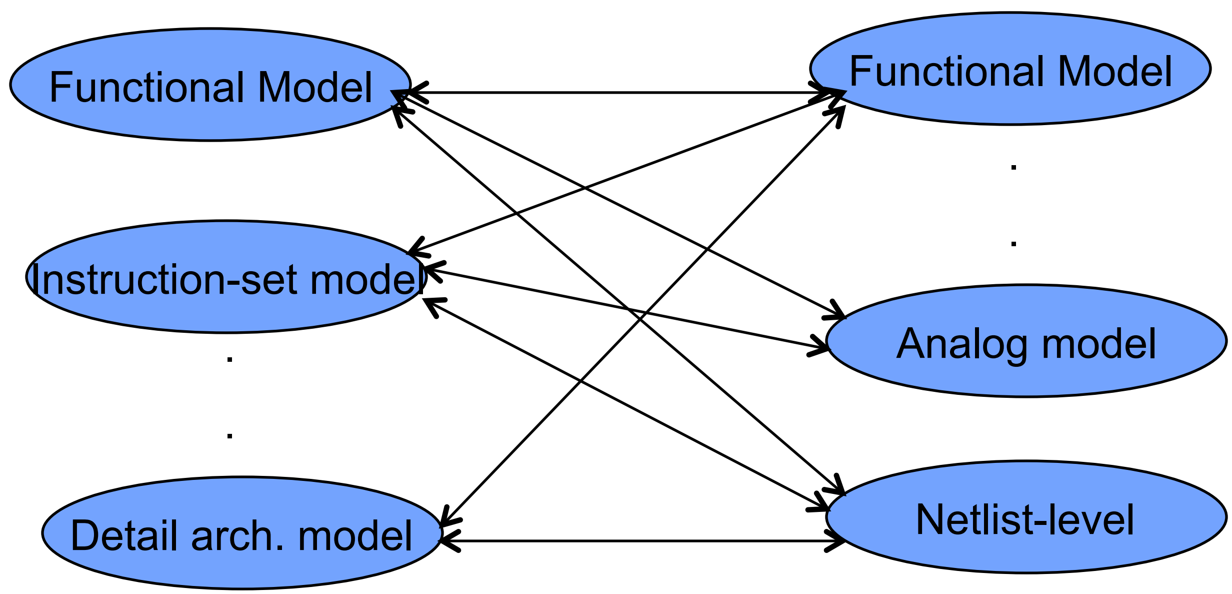


Synchronization and Time in cosimulation

- In case of a single simulator (say Verilog) there is no problem for timing as single event queue is managed for simulation.
- If there are several simulators and software programs in the domain:
  - hardware and software domain are using a handshaking protocol to keep their time (clock) synchronized. Signals (events) transferred from one side to the other should have attached a time stamp.
  - It is possible to use a loosely coupled strategy which allows the two domain to proceed more independently. If a signal is received with a time stamp lower than the current clock in the respective domain, the respective simulator have to be back up.

Aspects of cosimulation

- A frame work of cosimulation consists of variety of components, levels of abstractions and different models.



Final issues

- Come by my office hours (right after class)
- Any questions or concerns?